Multi-access coded caching: gains beyond cache-redundancy

Berksan Serbetci, Emanuele Parrinello and Petros Elia
Communication Systems Department, EURECOM, Sophia Antipolis, France
Email: {serbetci,parrinel,elia}@eurecom.fr

***Abstract*— The work considers the K-user cache-aided shared-link broadcast channel where each user has access to exactly z caches of normalized size γ, and where each cache assists exactly z users. For this setting, for two opposing memory regimes, we propose novel caching and coded delivery schemes which maximize the local caching gain, and achieve a *coding gain* larger than $1+K\gamma$ (users served at a time) despite the fact that the total cache redundancy remains $K\gamma$ irrespective of z. Interestingly, when $z = \frac{K-1}{K\gamma}$, the derived optimal coding gain is $K\gamma z + 1$, matching the performance of a hypothetical scenario where each user has its own dedicated cache of size $z\gamma$.**

I. Introduction

A significant step toward understanding the fundamental limits of interference-limited caching networks was achieved in the work of Maddah-Ali and Niesen in [1] where an information theoretic study of the cache-aided broadcast channel (BC) revealed large improvements over the conventional caching approaches, as a result of a carefully designed cache placement phase which allowed a simultaneous delivery to many users at a time. In particular, the work in [1] considered a setting of a single server having access to a library of N unit-sized files and connected via a shared-link unit-capacity BC to K receiving users, where each user has access to its own dedicated cache of size $M \leq N$. In this setting, [1] showed that any possible set of requests by the K users, can be served with delay $T = \frac{K(1-\gamma)}{K\gamma+1}$, where $\gamma = M/N$ is the normalized cache size. This revealed an ability to serve $K\gamma + 1$ users at a time, where this number is commonly referred to as the *coding gain* or simply the *degrees-of-freedom (DoF)*. For this setting, the gain was proven in [2], [3] to be optimal under the constraint of uncoded cache placement. The coded caching ideas sparked considerable interest, resulting in a variety of related works that include [5], [6], [7], [8], [9], [10].

A. Coded caching with multiple-access

Most of the works on coded caching consider scenarios where each user has its own dedicated cache. However in a variety of settings, such as different cellular networks, users can conceivably connect to multiple caches whose coverage areas may overlap. This consideration motivated the work in [11] which considered a similar K-user shared-link BC, where each user is assisted by exactly $z > 1$ caches, and where each cache can serve exactly z users. In this context, the work in [11] provided a caching and delivery strategy whose centralized variant[1] achieves a worst-case delivery time of

$$T = \frac{K(1-z\gamma)}{K\gamma+1} \tag{1}$$

reflecting an ability to increase the local caching gain to $z\gamma$ (meaning that each user sees a fraction $z\gamma$ of each file), as well as an ability to preserve the coding gain to $K\gamma + 1$, by virtue of a scheme that served $K\gamma + 1$ users at a time. For this same setting, in [12] the authors provide explicit designs for $K = 4$, $N = 4$, $z = \{2, 3\}$, $M = 1$ and $K = 6$, $N = 6$, $z = 3$, $M = 1$, for which matching lower bounds are also developed to prove optimality of the schemes under uncoded cache placement. Finally, the authors provide a scheme for the extreme case of $z = K - 1$.

For this multi-access setting, we will show that the coding gain $\frac{K(1-z\gamma)}{T}$ can exceed $K\gamma + 1$. In particular, for two opposing memory regimes, we propose two novel schemes which can serve, on average, more than $K\gamma + 1$ users at a time. For the special case of $z = \frac{K-1}{K\gamma}$, the achieved gain is proven to be optimal under uncoded cache placement.

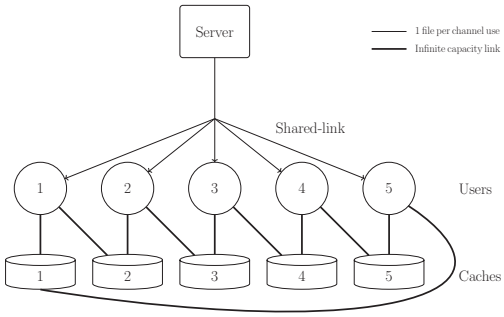

Fig. 1. Setting for $K = 5$ and $z = 2$.

II. System model and problem definition

We consider a network where K users are connected via an error-free shared link to a server storing N ($N \geq K$) files $W^1, W^2, \ldots, W^N$. Each user has access to z out of K helper

This work was supported by the European Research Council under the EU Horizon 2020 research and innovation program / ERC grant agreement no. 725929.

[1]The work in [11] proposed a decentralized (stochastic) cache placement scheme, whose performance is slightly reduced over the easy-to-extend-to centralized variant whose delivery time we recorded above.

$$X_1 = \sum_{k=1}^{K} \left[\sum_{j=1}^{\min\{k-z-1,z-1\}} \left[\left\lfloor \frac{K-(k+z+j)}{2} \right\rfloor + 1 \right]_+ + \sum_{j=\max\{1,k-z\}}^{z-1} \left[\left\lfloor \frac{K-2z-2j}{2} \right\rfloor + 1 \right]_+ \right],$$

$$X_2 = K \left[\frac{\left\lfloor \frac{K-2-z}{3} \right\rfloor - z}{2} + 1 \right]_+ \left(K-4z+3-3 \left[\frac{\left\lfloor \frac{K-2-z}{3} \right\rfloor - z}{2} + 1 \right]_+ \right), \qquad S = \frac{K(K-2z+2)}{4}. \quad (2)$$

caches[2], each of size $M = N\gamma$ (units of file), where $\gamma \in \{\frac{1}{K}, \frac{2}{K}, \ldots, 1\}$. We will use $\mathcal{Z}_k$ to denote the content in cache k, and we will assume that each user has an unlimited capacity link to the caches it is connected to. Reflecting the assumption that each user has access to exactly z caches and each cache connects to exactly z users, we will consider, without loss of generality, the symmetric topology where each user $k \in [K] \triangleq \{1, 2, \ldots, K\}$ is associated to caches

$$\mathcal{C}_k \triangleq \langle k, k+1, \ldots, k+z-1 \rangle \subseteq [K]$$

where in the above we use the notation $\langle \mathcal{M} \rangle = \{m \mid m \text{ for } m \leq K \; ; \; m-K \text{ for } m > K, \; m \in \mathbb{Z}^+, \forall m \in \mathcal{M}\}$. A pictorial representation of the studied setting can be found in Figure 1.

The system works in two phases: the cache placement phase and the delivery phase. The first consists of filling the caches with the content of the library without knowledge of the users' demands. In the delivery phase, each user k requests a file from the library. We denote the index of such file by d_k and we collect all the indices of the requested files in the demand vector $\boldsymbol{d} = (d_1, d_2, \ldots, d_K)$. The work focuses on the worst case where all users request different files. Upon reception of the demand vector $\boldsymbol{d}$, the server will transmit a message X that spans T units of time. Each user k will use X and their own available cache content $\cup_{i \in \mathcal{C}_k} \mathcal{Z}_i$ to decode the desired file W^{d_k}. Our objective is to provide a caching and delivery scheme that reduces the delivery delay T.

III. Main Results

This section presents the main results. The proof of the following theorem follows from the scheme described in Section IV.

Theorem 1. *In the coded caching setting where each user is connected to z caches, when $K\gamma = 2$, the delivery time*

$$T = \frac{X_1 + X_2}{S} \quad (3)$$

is achievable [3], where X_1, X_2 and S are given in (2).

[2]Notice that the assumption of having as many users as caches can be relaxed to a more general case where the system has more users than caches, with a potential non uniform distribution of the users among the caches. Under these circumstances, the schemes described here can be combined with the non-uniformity adaptive coded caching scheme presented in [7] for the single transmitter setting.

[3]The above expression holds for the case where $S(K-2z) - 4X_1$ is non negative and divisible by 3. This assumption can be removed at a small cost of increased scheme subpacketization.

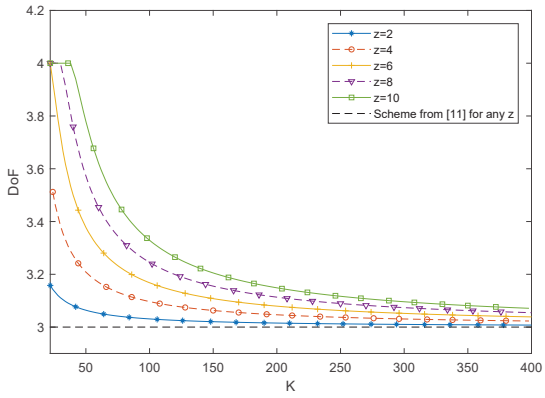

Fig. 2. DoF as a function of K for several z. Case of $K\gamma = 2$.

Remark 1. The scheme consists of transmissions serving either $K\gamma + 1 = 3$ users or 4 users at a time. Figure 2 shows, for several values of z, the achievable DoF as a function of K. We can see how the DoF always exceeds $K\gamma + 1 = 3$ and can approach 4 for some values of K and z.

We now characterize the optimal worst-case delivery time (under the assumption of uncoded cache placement) for the case of $z = \frac{K-1}{K\gamma}$, for any $K\gamma$.

Theorem 2. *In the addressed K-user caching network with access to an integer number $z = \frac{K-1}{K\gamma}$ of caches of normalized size γ, the optimal delivery time, under the assumption of uncoded cache placement, takes the form*

$$T^* = \frac{K(1-\gamma z)}{K\gamma z + 1} = \frac{1}{K} \quad (4)$$

corresponding to a DoF of $K\gamma z + 1$ users served at a time.

The scheme that achieves the above performance is described in Section V. The optimality — under uncoded cache placement — follows directly from the fact that the achieved performance matches the optimal performance (cf. [2],[3]) of a dedicated-cache coded caching setting (identical to that in [1]), where each cache has an augmented size equal to $z\gamma$.

IV. Caching and Delivery Scheme for $K\gamma = 2$

In this section we present a caching and delivery scheme for the case of $K\gamma = 2$. The scheme preserves the full local caching gain as in [11], and achieves a coding gain that strictly exceeds $K\gamma + 1$.

A. Cache placement algorithm

In the cache placement phase, we first split each file W^n into $S = \frac{K(K-2z+2)}{4}$ subfiles $W^n_{\mathcal{T}}$ for each pair $\mathcal{T} \triangleq \{\mathcal{T}_1, \mathcal{T}_2\}$ from the set

$$\Psi \triangleq \{\mathcal{T} : \mathcal{T}_1 \in [K-z], \mathcal{T}_2 \in [\mathcal{T}_1+z : 2 : \min\{K-z+\mathcal{T}_1, K\}]\} \tag{5}$$

where in the above we used the notation $[a : b : c]$ to denote an ordered set of integers[4], from a to c, in additive steps of b. After splitting the files, each cache k is filled as follows

$$\mathcal{Z}_k = \{W^n_{\mathcal{T}} \mid \forall n \in [N], \forall \mathcal{T} \ni k\}. \tag{6}$$

a) Verifying the memory constraint: To show that the cache placement satisfies the per-cache memory $M = \frac{2N}{K}$, we focus without loss of generality on cache 1, and note that the number of subfiles (per file) in this cache is $\frac{K-2z}{2} + 1$. Recalling that each such subfile is of size $1/S$, yields

$$\frac{N\left(\frac{K-2z}{2}+1\right)}{\frac{K(K-2z+2)}{4}} = \frac{2N}{K} = M$$

thus proving that the memory constraint is satisfied.

B. Delivery scheme

The delivery scheme has two phases, where the first phase transmits XORs composed of 4 subfiles, while the second phase transmits XORs composed of $K\gamma + 1 = 3$ subfiles.

1) Phase 1: Recall from above that any subfile $W^{d_k}_{\mathcal{T}}$, $k \in \mathcal{T} \in \Psi$ ($\mathcal{T}_1 \in [k : 1 : \langle k+z-1\rangle]$), is already available at one of the caches seen by the requesting user $k \in [K]$. For each user $k \in [K]$, the aim of this first phase is to serve the subfiles in the set

$$\Big\{W^{d_k}_{\mathcal{T}_1,\mathcal{T}_2} \mid \forall\{\mathcal{T}_1,\mathcal{T}_2\} \in \Psi : \mathcal{T}_1 \in \langle[k-z+1{:}1{:}k-1]\rangle \cup \langle[k+z{:}1{:}k+2z-2]\rangle\Big\}. \tag{7}$$

For any $k \in [K]$, let us define the following two sets

$$\begin{aligned}\Omega_{k,1} &\triangleq [k+1 : 1 : k+z-1]\\ \Omega_{k,2} &\triangleq [k-z+1 : 1 : k-1]\end{aligned} \tag{8}$$

and the set

$$B_{k,j} \triangleq [\Omega_{k,1}(j) + z : 2 : U_{k,j}] \tag{9}$$

where

$$U_{k,j} \triangleq \begin{cases} \langle \Omega_{k,2}(j) - z\rangle, & \text{if } \Omega_{k,2}(j) - z < 0\\ K, & \text{otherwise.}\end{cases} \tag{10}$$

and where in the above we used the notation $\Gamma(j)$ to denote the j-th element of an ordered set Γ. Next, for any $k \in [K]$ and any $j \in [z-1]$ we form the following XOR

$$\begin{aligned}X(k,j,m) = W^{d_{\langle k-z+1\rangle}}_{\langle\Omega_{k,1}(j)\rangle, B_{k,j}(m)} \oplus W^{d_k}_{\langle\Omega_{k,2}(j)\rangle, B_{k,j}(m)}\\ \oplus W^{d_{\langle B_{k,j}(m)-z+1\rangle}}_{\langle\Omega_{B_{k,j}(m),1}(j)\rangle,k} \oplus W^{d_{B_{k,j}(m)}}_{\langle\Omega_{B_{j,k}(m),2}(j)\rangle,k}.\end{aligned} \tag{11}$$

[4]Note that b may be less than zero.

Creating the above XORs for every $m \in [|B_{k,j}|]$ and every $k \in [K]$, spans the entire set of requested files in (7), and what we show below is that each component subfile (in the XORs) can be successfully decoded by its corresponding user.

a) Decoding: Consider any XOR as in (11) and let us focus on the subfiles $W^{d_{\langle k-z+1\rangle}}_{\Omega_{k,1}(j),B_{k,j}(m)}$ and $W^{d_k}_{\Omega_{k,2}(j),B_{k,j}(m)}$ which are desired by users $\langle k-z+1\rangle$ and k, respectively. By the cache placement phase, we notice that the subfiles $W^{d_{\langle B_{k,j}(m)-z+1\rangle}}_{\mathcal{T}_{B_{k,j}(m),z,1}(j),k}$ and $W^{d_{B_{k,j}(m)}}_{\mathcal{T}_{b_{j,k,z}(m),z,2}(j),k}$ are both cached in cache k, thus enabling both users $\langle k-z+1\rangle$ and k to subtract these subfiles from $X(k,j,m)$. Next, we also notice that user $\langle k-z+1\rangle$ can cache out $W^{d_k}_{\Omega_{k,2}(j),B_{k,j}(m)}$ since $\Omega_{k,2}(j) \in \cup_{i\in\mathcal{C}_{\langle k-z+1\rangle}}\mathcal{Z}_i$. Similarly, user k can remove $W^{d_{\langle k-z+1\rangle}}_{\Omega_{k,1}(j),B_{k,j}(m)}$ from $X(k,j,m)$ because $\Omega_{k,1}(j) \in \cup_{i\in\mathcal{C}_k}\mathcal{Z}_i$. Hence, we conclude that any XOR in (11) is decodable by both users k and $\langle k-z+1\rangle$. In the same way, it can be shown that users $\langle B_{k,j}(m)-z+1\rangle$ and $B_{k,j}(m)$ can successfully decode their own requested subfiles.

2) Phase 2: We start by defining the set

$$\delta = \left[z : 2 : \left\lfloor\frac{K-2-z}{3}\right\rfloor\right]$$

as well as the following set of triplets

$$\Theta = \Bigg\{(\theta_1,\theta_2,\theta_3)|\theta_1 = \delta(j), \theta_2 = 2\delta(j)+z+2(i-1),$$
$$\theta_3 = \delta(j)+2(i-1), \quad j \in [|\delta|],\ i \in \left[\frac{K-3\delta(j)-z}{2}\right]\Bigg\}. \tag{12}$$

For each triplet $\theta \in \Theta$ and for each $p \in [K]$, we generate the following two XORs

$$\begin{aligned}Y_p(\theta,1) = W^{d_p}_{\langle\theta_2-\theta_1+p-1\rangle,\langle\theta_2+p-1\rangle} \oplus W^{d_{\langle\theta_2-\theta_1-z+p\rangle}}_{p,\langle\theta_2+p\rangle}\\ \oplus W^{d_{\langle\theta_2+p-1\rangle}}_{\langle z-2+p\rangle,\langle z-2+\theta_3+p\rangle}\end{aligned} \tag{13}$$

$$\begin{aligned}Y_p(\theta,2) = W^{d_p}_{\langle\theta_2-\theta_1+p\rangle,\langle\theta_2+p\rangle} \oplus W^{d_{\langle\theta_2-\theta_1-z+1+p\rangle}}_{p,\langle\theta_2+p\rangle}\\ \oplus W^{d_{\langle\theta_2+p-1\rangle}}_{\langle z+p-1\rangle,\langle z+\theta_3+p-1\rangle}.\end{aligned} \tag{14}$$

It can be shown (this is not done here due to lack of space) that, due to the structure of the XORs, as we go over all $\theta \in \Theta, p \in [K]$, no subfile is ever repeated. This allows us to conclude that the two phases successfully include all desired subfiles by all the users. As we did for phase 1, below we show that each subfile in the above XORs from (13) and (14) can be decoded successfully by their requesting user.

a) Decoding: For any $p \in [K]$, we will prove that the subfiles in $Y_p(\theta,1)$ can be decoded by their intended users. The decodability proof for $Y_p(\theta,2)$ will then follow directly.

User p can cache out subfiles $W^{d_{\langle\theta_2-\theta_1-z+p\rangle}}_{p,\langle p+\theta_2\rangle}$ and $W^{d_{\langle\theta_2+p-1\rangle}}_{\langle z-2+p\rangle,\langle z-2+\theta_3+p\rangle}$ from $Y_p(\theta,1)$ since their subscripts $p \in \cup_{i\in\mathcal{C}_p}\mathcal{Z}_i$ and $\langle z-2+p\rangle \in \cup_{i\in\mathcal{C}_p}\mathcal{Z}_i$ correspond to the caches that user p is connected to. Next, we notice that user

$\langle \theta_2 - \theta_1 - z + p \rangle$ is connected to cache $\langle \theta_2 - \theta_1 + p - 1 \rangle$ and thus it can cache out subfile $W^{d_p}_{\langle \theta_2-\theta_1+p-1\rangle,\langle \theta_2+p-1\rangle}$. The same user $\langle \theta_2-\theta_1-z+p\rangle = \langle \delta(j)+p+2(i-1)\rangle$ has access to subfiles with subscripts in the set $\langle [\delta(j)+p+2(i-1):1:\delta(j)+p+2(i-1)+z-1]\rangle$. A relabelling of $\langle \theta_3+p+z-2\rangle$ to $\langle \delta(j)+z+p-2+2(i-1)\rangle$ highlights that user $\langle \theta_2-\theta_1-z+p\rangle$ can also remove $W^{d_{\langle \theta_2+p-1\rangle}}_{\langle z-2+p\rangle,\langle z-2+\theta_3+p\rangle}$ from $Y_p(\theta,1)$, and hence obtains its desired subfile $W^{d_{\langle \theta_2-\theta_1-z+p\rangle}}_{p,\langle \theta_2+p\rangle}$ successfully. Finally, we recall that user $\langle \theta_2+p-1\rangle$ has access to caches $\mathcal{C}_{\langle \theta_2+p-1\rangle} = \langle [\theta_2+p-1:1:\theta_2+p-2+z]\rangle$ and hence can successfully decode its desired subfile since it can cache out subfiles $W^{d_p}_{\langle \theta_2-\theta_1+p-1\rangle,\langle \theta_2+p-1\rangle}$ and $W^{d_{\langle \theta_2-\theta_1-z+p\rangle}}_{p,\langle \theta_2+p\rangle}$.

3) Performance of the algorithm: We observe that each generated XOR serves a different set of 3 or 4 subfiles, which can all be decoded. It can also be shown that the proposed delivery scheme successfully satisfies any demand vector $\boldsymbol{d}$. The proof is easy and is here omitted due to lack of space. Following the construction, we can readily count the total number of XORs transmitted during phase 1 and phase 2 to respectively be X_1 and X_2 from (2), which concludes the proof of the achievable delay in Theorem 1.

C. Example

In this subsection we offer an example that may help to better understand the scheme. We consider the setting with parameters $K = 10$, $K\gamma = 2$, and $z = 2$. For the sake of simplicity, we will use 0 to represent the index 10 and also, when describing a double index i, j, we will omit the comma.

In the placement phase, we first split each file, according to (5), into $S = 20$ equally-sized subfiles with indices

$$\Psi = \{13, 15, 17, 19, 24, 26, 28, 20, 35, 37, 39, 46, 48, 40, 57, 59, 68, 60, 79, 80\}$$

and we then fill the caches according to (6) as follows

$$\begin{aligned}
\mathcal{Z}_1 &= \{W^n_{13}, W^n_{15}, W^n_{17}, W^n_{19},\ \forall n \in [N]\}\\
\mathcal{Z}_2 &= \{W^n_{24}, W^n_{26}, W^n_{28}, W^n_{20},\ \forall n \in [N]\}\\
&\vdots\\
\mathcal{Z}_9 &= \{W^n_{19}, W^n_{39}, W^n_{59}, W^n_{79},\ \forall n \in [N]\}\\
\mathcal{Z}_0 &= \{W^n_{20}, W^n_{40}, W^n_{60}, W^n_{80},\ \forall n \in [N]\}.
\end{aligned}$$

We notice that the cache placement guarantees an empty intersection $\mathcal{Z}_k \cap \mathcal{Z}_{\langle k+1\rangle} = \emptyset$ of any $z = 2$ neighboring caches, and thus a full local caching gain ($z\gamma = 0.4$).

In the delivery phase we consider the worst-case demand vector $\boldsymbol{d} = (1, 2, \ldots, 9, 0)$. We will list the XORs of phase 1 and phase 2, but before doing that, let us offer some intuition on the design of the XORs of the first phase.

Let us consider a pair of users (say, users 0 and 1) that "see" a common cache (in this case, cache 1). For these two users we will create a generic XOR

$$W^0_{\sigma_1,\sigma_2} \oplus W^1_{\tilde{\sigma}_1,\tilde{\sigma}_2} \tag{15}$$

which will be combined with another XOR

$$W^3_{\tau_1,\tau_2} \oplus W^4_{\tilde{\tau}_1,\tilde{\tau}_2} \tag{16}$$

which is meant for another pair of users, say 3 and 4, that again share a common cache (cache 4). Combining the two XORs yields a new XOR $X = W^0_{\sigma_1,\sigma_2} \oplus W^1_{\tilde{\sigma}_1,\tilde{\sigma}_2} \oplus W^3_{\tau_1,\tau_2} \oplus W^4_{\tilde{\tau}_1,\tilde{\tau}_2}$ of 4 subfiles. To guarantee decoding for all, we will set $\sigma_1 = \tilde{\sigma}_1 = 4$ to let user 3 and user 4 "cache out" from X the subfiles in (15) and similarly we set $\tau_1 = \tilde{\tau}_1 = 1$ in order to let users 0 and 1 cache out the XOR in (16). Next, we choose $\sigma_2 = 2$ so that user 1 can remove subfile $W^0_{4,2}$ from X and $\tilde{\sigma}_2 = 0$ to let user 0 remove subfile $W^1_{4,0}$ from X. A similar choice of τ_2 and $\tilde{\tau}_2$ will result in[5]

$$X = W^0_{24} \oplus W^1_{40} \oplus W^3_{15} \oplus W^4_{13}. \tag{17}$$

The list of XORs sent during phase 1 is given below.

$$\begin{aligned}
X(1,1,1) &= W^0_{24} \oplus W^1_{40} \oplus W^3_{15} \oplus W^4_{13}\\
X(1,1,2) &= W^0_{26} \oplus W^1_{60} \oplus W^5_{17} \oplus W^6_{15}\\
X(1,1,3) &= W^0_{28} \oplus W^1_{80} \oplus W^7_{19} \oplus W^8_{17}\\
X(2,1,1) &= W^1_{35} \oplus W^2_{15} \oplus W^4_{26} \oplus W^5_{24}\\
X(2,1,2) &= W^1_{37} \oplus W^2_{17} \oplus W^6_{28} \oplus W^7_{26}\\
X(2,1,3) &= W^1_{39} \oplus W^2_{19} \oplus W^8_{20} \oplus W^9_{28}\\
X(3,1,1) &= W^2_{46} \oplus W^3_{26} \oplus W^5_{37} \oplus W^6_{35}\\
X(3,1,2) &= W^2_{48} \oplus W^3_{28} \oplus W^7_{39} \oplus W^8_{37}\\
X(3,1,3) &= W^2_{40} \oplus W^3_{20} \oplus W^9_{13} \oplus W^0_{39}\\
X(4,1,1) &= W^3_{57} \oplus W^4_{37} \oplus W^6_{48} \oplus W^7_{46}\\
X(4,1,2) &= W^3_{59} \oplus W^4_{39} \oplus W^8_{40} \oplus W^9_{48}\\
X(5,1,1) &= W^4_{68} \oplus W^5_{48} \oplus W^7_{59} \oplus W^8_{57}\\
X(5,1,2) &= W^4_{60} \oplus W^5_{40} \oplus W^9_{15} \oplus W^0_{59}\\
X(6,1,1) &= W^5_{79} \oplus W^6_{59} \oplus W^8_{60} \oplus W^9_{68}\\
X(7,1,1) &= W^6_{80} \oplus W^7_{60} \oplus W^9_{17} \oplus W^0_{79}.
\end{aligned}$$

In phase 2, the $X_2 = 20$ transmissions from (13) and (14) are cyclically generated as shown below.

$$\begin{aligned}
Y_1(\theta,1) &= W^1_{46} \oplus W^3_{17} \oplus W^6_{13},\\
Y_2(\theta,1) &= W^2_{57} \oplus W^4_{28} \oplus W^7_{24}\\
Y_3(\theta,1) &= W^3_{68} \oplus W^5_{39} \oplus W^8_{35}\\
&\vdots\\
Y_0(\theta,1) &= W^0_{35} \oplus W^2_{60} \oplus W^5_{20}
\end{aligned}$$

and

$$\begin{aligned}
Y_1(\theta,2) &= W^1_{57} \oplus W^4_{17} \oplus W^6_{24}\\
Y_2(\theta,2) &= W^2_{68} \oplus W^5_{28} \oplus W^7_{35}\\
Y_3(\theta,2) &= W^3_{79} \oplus W^6_{39} \oplus W^8_{46}\\
&\vdots\\
Y_0(\theta,2) &= W^0_{46} \oplus W^3_{60} \oplus W^5_{13}.
\end{aligned}$$

[5]This intuition can be generalized for $z > 2$ and used as a baseline in the general description of the scheme presented in Section IV-B1.

In the end, we have $S = 20$, $X_1 = 15$ and $X_2 = 20$ which gives

$$T = \frac{X_1 + X_2}{S} = \frac{35}{20}$$

and a coding gain $K(1 - z\gamma)/T = 3.43$.

V. Coded caching for $K = K\gamma z + 1$

Corresponding to Theorem 2, we now present the optimal caching and delivery scheme for $z = \frac{K-1}{K\gamma}$ for any $K\gamma$.

A. Cache placement algorithm

In the cache placement phase, each file W^n is first split into K subfiles W^n_ϕ, $\phi \in \Phi$ where each $K\gamma$-tuple $\phi \triangleq \{\phi_1, \phi_2, \ldots, \phi_{K\gamma}\}$ is drawn from the set

$$\Phi \triangleq \{\phi : \phi_1 \in [K], \phi_j = \langle \phi_{j-1} + z \rangle, \forall j \in [2 : K\gamma]\} \quad (18)$$

of size K. Then each cache k is filled as follows

$$\mathcal{Z}_k = \{W^n_\Phi \mid \forall n \in [N], \forall \Phi \ni k\} \quad (19)$$

forcing each integer $k \in [K]$ to appear in Φ exactly $K\gamma$ times, thus guaranteeing that each cache stores exactly $K\gamma$ subfiles from each file, thus respecting the cache-size constraint.

What the above placement also guarantees is that, by construction of the set Φ, all subfiles of each file stored in any z consecutive caches, are different. This is due to the fact that any two elements of each $K\gamma$-tuple $\phi \in \Phi$ have distance at least z. Thus, each user k has access to $K\gamma z = K\frac{K-1}{Kz}z = K-1$ different subfiles of its requested file W^{d_k}. We denote by $W^{d_k}_{\rho_k}$ the one remaining subfile desired by user k, for a specific $K\gamma$-tuple $\rho_k = \Phi \setminus \{\cup_{i \in \mathcal{C}_k} \mathcal{Z}_i\}$.

B. Delivery and decoding

Upon reception of the demand vector $\boldsymbol{d}$, the server multicasts a single XOR

$$X = \bigoplus_{k \in [K]} W^{d_k}_{\rho_k} \quad (20)$$

to all users of the network. By virtue of the fact that each user is only missing a single subfile, we can deduce that each user k can cache out from X all $K - 1$ subfiles $\{W^{d_j}_{\rho_j}\}_{j \in [K] \setminus \{k\}}$ to successfully decode its own requested subfile $W^{d_k}_{\rho_k}$. Consequently the total delivery time is naturally equal to $T = |X| = \frac{1}{K}$.

C. Example

Let us consider the case of $K\gamma = 3$, $z = 2$ and $K = 7$. We first split each file into 7 equally sized subfiles with indices

$$\Phi = \{135, 136, 146, 246, 247, 257, 357\}.$$

and fill each cache, according to (19), as follows

$$\mathcal{Z}_1 = \{W^n_{135}, W^n_{136}, W^n_{146}, \ \forall n \in [N]\}$$
$$\mathcal{Z}_2 = \{W^n_{246}, W^n_{247}, W^n_{257}, \ \forall n \in [N]\}$$
$$\mathcal{Z}_3 = \{W^n_{135}, W^n_{136}, W^n_{357}, \ \forall n \in [N]\}$$
$$\mathcal{Z}_4 = \{W^n_{146}, W^n_{246}, W^n_{247}, \ \forall n \in [N]\}$$
$$\mathcal{Z}_5 = \{W^n_{135}, W^n_{257}, W^n_{357}, \ \forall n \in [N]\}$$
$$\mathcal{Z}_6 = \{W^n_{136}, W^n_{146}, W^n_{246}, \ \forall n \in [N]\}$$
$$\mathcal{Z}_7 = \{W^n_{247}, W^n_{257}, W^n_{357}, \ \forall n \in [N]\}.$$

In the delivery phase we consider the delivery vector $\boldsymbol{d} = (1, 2, \ldots, 7)$. We notice that the placement and topology jointly guarantee that each user is missing only a single subfile. For example, user 1 is only missing subfile W^1_{357}. Placing all these missing subfiles together, the server sends

$$X = W^1_{357} \oplus W^2_{146} \oplus W^3_{257} \oplus W^4_{136} \oplus W^5_{247} \oplus W^6_{135} \oplus W^7_{246} \quad (21)$$

which guarantees that each user can cache out exactly 6 elements to decode their own subfile. The delay is $T = \frac{1}{7}$ and the DoF of $K\gamma z + 1 = 7$.

VI. Conclusion

For the multiple access coded caching problem, we have proposed novel coded caching schemes that achieve a coding gain that exceeds $K\gamma + 1$. To the best of our knowledge, in the context of worst-case delivery time and for $N \geq K$, this is the first instance of a fully-connected shared-link problem that experiences a gain larger than $K\gamma+1$. An interesting extension of this work is its generalization to all possible values of $K\gamma$.

References

[1] M. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, May 2014.

[2] K. Wan, D. Tuninetti, and P. Piantanida, "On the optimality of uncoded cache placement," in *Information Theory Workshop (ITW), IEEE*, 2016.

[3] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "The exact rate-memory tradeoff for caching with uncoded prefetching," *IEEE Trans. Inf. Theory*, Feb 2017.

[4] E. Lampiris, J. Zhang, and P. Elia, "Cache-aided cooperation with no CSIT," in *Proc. IEEE Int. Symp. on Inform. Theory (ISIT)*, June 2017.

[5] T. X. Vu, S. Chatzinotas, and B. Ottersten, "Coded caching and storage planning in heterogeneous networks," in *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, March 2017.

[6] K. Wan, D. Tuninetti, M. Ji, and G. Caire, "Novel inter-file coded placement and d2d delivery for a cache-aided fog-ran architecture," 2018. [Online]. Available: https://arxiv.org/pdf/1811.05498.pdf

[7] E. Parrinello, A. Ünsal, and P. Elia, "Fundamental limits of caching in heterogeneous networks with uncoded prefetching," *arXiv preprint https://arxiv.org/pdf/1811.06247.pdf*, 2018.

[8] N. Zhang and M. Tao, "Fitness-aware coded multicasting for decentralized caching with finite file packetization," *IEEE Wireless Communications Letters*, Oct 2018.

[9] H. Ghasemi and A. Ramamoorthy, "Algorithms for asynchronous coded caching," in *51st Asilomar Conference on Signals, Systems, and Computers*, Oct 2017.

[10] E. Ozfatura, T. Rarris, D. Gndz, and O. Ercetin, "Delay-aware coded caching for mobile users," in *IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sep. 2018.

[11] J. Hachem, N. Karamchandani, and S. Diggavi, "Coded caching for multi-level popularity and access," *IEEE Trans. Inf. Theory*, May 2017.

[12] K. S. Reddy and N. Karamchandani, "On the exact rate-memory trade-off for multi-access coded caching with uncoded placement," in *2018 International Conference on Signal Processing and Communications (SPCOM)*, July 2018.

[13] E. Lampiris and P. Elia, "Achieving full multiplexing and unbounded caching gains with bounded feedback resources," in *Proc. IEEE Int. Symp. on Inform. Theory (ISIT)*, June 2018.

[14] E. Lampiris and P. Elia, "Full coded caching gains for cache-less users," in *Information Theory Workshop (ITW), IEEE*, 2018.